

Objective-C Fundamentals

1DevDay, 2011

Priya Rajagopal

Jeff Kelley

What is Objective-C

- Superset of C
- Extends with Object Oriented Constructs
- Dynamic

Object

- Composed of data and associated methods
- New Data Type : “id”
- “is-a” variable
 - Introspection on objects
 - What is your Class?

Class

- Blueprint for objects
- Also an Object in Obj-C (!)
 - Data Type “Class”
 - Class Methods (“+”)
 - Instance Methods (“-”)
- Class Interface in .h file
 - Method declarations
 - Instance variables declarations
 - Properties declarations
- Class Implementation in .m file
 - Implementation of methods
- NSObject
 - Root Class

Method Declaration

```
- (void)handleServerCommand:(NSString*)  
command length: (NSInteger)commandLength;
```

Method
Type

Return
Type

Method
Name

Parameter
Types

Parameter
Names

Method Invocation: Messaging

```
[myObject handleServerCommand:buffer  
length:len];
```

Receiver

Message

Properties

Convenience Notation for-

- Declaration of Accessor Methods

```
@property (attributes) type propName;
```

- Implementation of Accessor Methods

```
@synthesize propName;
```

Properties Declaration Details

```
@property (attributes) type propName;
```



```
-(void) setPropName :(type) newPropName ;  
-(type) propName;
```

Property Attributes

atomic	readwrite	assign
nonatomic	readonly	retain

Property Storage

- Manual declaration
 - To store property *myProperty* in instance variable *_myProperty*, use:
`@synthesize myProperty = _myProperty;`
- Automatic Declaration
`@synthesize myProperty;`
- If the instance variable is not declared in the object interface, the compiler adds it

Protocols

- “Protocol is simply a list of method declarations, unattached to a class definition”

```
@protocol <protocol name>  
    <method declarations>  
@end
```

- Independent of any class hierarchy
- Class that “adopts” / “conforms to” a protocol implements required protocol methods

```
@interface helperClass :  
    NSObject<protocol name>
```

Protocols: Define Methods that others implement

- The Class Interface specifies what methods the class implements. The protocol specifies what it expects to use.
- “Delegate” the responsibility of method implementation to other object

```
if ( [delegateObj conformsToProtocol:@protocol
(protocolName)] ) {
    [delegateObj protocolMethod];
....
}
```

Categories

- Alternate to sub-classing
- Add methods to an existing class
- Cannot add instance variables
- Inheriting classes will inherit the category methods as well
- Logical grouping of related functions

```
@interface ClassThatIsExtended ( CategoryName )  
// method declarations  
@end
```

Demo